Available online at www.sciencedirect.com
ScienceDirect
Theoretical Computer Science 372 (2007) 69–80
Theoretical Computer Science
www.elsevier.com/locate/tcs

Optimal semi-online algorithms for machine covering

Zhiyi Tan*, Yong Wu

Department of Mathematics, State Key Lab of CAD & CG, Zhejiang University, Hangzhou 310027, PR China

Received 27 April 2005; received in revised form 27 October 2006; accepted 19 November 2006

Communicated by D.-Z. Du

Abstract

This paper investigates the semi-online machine covering problems on $m \geq 3$ parallel identical machines. Three different semi-online versions are studied and optimal algorithms are proposed. We prove that if the total processing time of all jobs or the largest processing time of all jobs is known in advance, the competitive ratios of the optimal algorithms are both $m - 1$. If the total processing time and the largest processing time of all jobs are both known in advance, the competitive ratios of the optimal algorithms are $\frac{3}{2}$ when $m = 3$, and $m - 2$ when $m \geq 4$.

Keywords: Scheduling; Design and analysis of algorithms; Semi-online; Competitive ratio

1. Introduction

In this paper, we study the semi-online machine covering problem on m parallel identical machines. This problem was first proposed by Deuermeyer et al. [5] and has applications in the sequencing of maintenance actions for modular gas turbine aircraft engines. The problem can also be described with the following standard scheduling terminology. We are given a set $\mathcal{J}$ of independent jobs with positive processing times $p_1, p_2, \ldots, p_n$ and a set $\mathcal{M}$ of m parallel identical machines $M_1, M_2, \ldots, M_m$. We identify the jobs with their processing times. The goal is to assign the jobs to the machines so as to maximize the minimum machine load, where the *load* of a machine is the sum of the processing time of the jobs assigned to the machine. We denote this problem by $Pm||C_{\min}$.

A scheduling problem is called *online* if jobs arrive one by one, and we are required to schedule jobs irrevocably on machines as soon as they are given, without any knowledge of the successive jobs. If we have full information of job sequence before constructing a schedule, the problem is called *offline*. In fact, sometimes problems are neither strictly online nor offline, but somewhat in between. This means that some partial information about the jobs is available before constructing a schedule. Such a case is defined as a *semi-online* problem. Different partial information gives rise to different semi-online problems. Algorithms for online (semi-online) problems are called online (semi-online) algorithms.

The quality of the performance of an online (semi-online) algorithm is measured by its *competitive ratio*. For an instance $(\mathcal{M}, \mathcal{J})$ and an algorithm A, let $C^A(\mathcal{M}, \mathcal{J})$ (or shortly C^A) be the objective value produced by A and let

* Corresponding author. Tel.: +86 571 87951429; fax: +86 571 87953715.
E-mail address: tanzy@zju.edu.cn (Z. Tan).

0304-3975/$ - see front matter
doi:10.1016/j.tcs.2006.11.015

$C^*(\mathcal{M}, \mathcal{J})$ (or shortly C^*) be the optimal value in an offline version. Then the competitive ratio of A is defined as the smallest number c such that for any $(\mathcal{M}, \mathcal{J})$, $C^*(\mathcal{M}, \mathcal{J}) \leq cC^A(\mathcal{M}, \mathcal{J})$. An online (semi-online) scheduling problem has a *lower bound* ρ if there is no online (semi-online) algorithm with a competitive ratio smaller than ρ. An online (semi-online) algorithm A is called *optimal* if its competitive ratio matches the lower bound of the problem.

Several semi-online variants have been studied so far. For example, Azar and Regev [2] considered those problems whose optimal objective function value C^* is known in advance (denoted by *opt*). Kellerer et al. [12] considered the problem that the total processing time of all the jobs $T = \sum_{j=1}^{n} p_j$ is known in advance (denoted by *sum*). He and Zhang [11] considered the problem that the largest processing time of all jobs $p_{\max} = \max_{j=1,\ldots,n} p_j$ is known in advance (denoted by *max*). In the same paper, they also considered the problem that the processing time of jobs are tightly-grouped, i.e. $p_j \in [p, rp]$, $j = 1, \ldots, n$, where $p > 0$ and $r \geq 1$ (denoted by *tightly-grouped*). Seiden et al. [13] considered the information that jobs are arriving in non-increasing processing time order, i.e. $p_1 \geq p_2 \geq \cdots \geq p_n$ (denoted by *decr*). In semi-online studies, it is important to derive respective optimal semi-online algorithms for different semi-online problems. Comparing their competitive ratios with that of corresponding optimal online algorithms, we can see which type of partial information, and to what extent, can improve the performance of a semi-online algorithm.

For the online version of $Pm||C_{\min}$, Woeginger [15] showed that LS is the optimal algorithm with competitive ratio m, where LS always assigns current job to the machine with the smallest current load. He [9] proved that LS is still optimal for $Pm|tightly\text{-}grouped|C_{\min}$ with competitive ratio r if $1 \leq r \leq m$, and m otherwise. For $Pm|decr|C_{\min}$, LS has a competitive ratio $\frac{4m-2}{3m-1}$ [4] and is optimal when $m = 2, 3$ [10]. Azar and Epstein [1] proposed an algorithm $FILL$ for $Pm|opt|C_{\min}$ with competitive ratio $2 - \frac{1}{m}$, and it is optimal for $m = 2, 3, 4$.

In this paper, we will consider semi-online versions of $Pm||C_{\min}$ with the total processing time of all jobs T and the largest processing time of all jobs $p_{\max}$ are known in advance. Since optimal algorithms for two machine cases were given in [8], we focus on the case of $m \geq 3$. We will present optimal algorithms for $Pm|sum|C_{\min}$ and $Pm|max|C_{min}$, respectively. The competitive ratios of two algorithms are both $m - 1$, which is a little smaller than that of LS. To further shed light on useful of different types of information, we are interested in whether a combination of two types of information can admit construction of a semi-online algorithm with smaller competitive ratio than that of the case where only one type of information is available [14,6]. Therefore, semi-online problems where both T and $p_{\max}$ are known in advance are also considered. It is not difficult to verify that the algorithm SM given in [14] is the optimal algorithm for $P2|sum\ \&\ max|C_{\min}$ with competitive ratio 5/4. In this paper, we will give optimal algorithms for $m \geq 3$. The competitive ratios of the optimal algorithms are 3/2 if $m = 3$, and $m - 2$ if $m \geq 4$, which is smaller than that for only one kind of partial information known in advance.

It is believed in the literature that two types of partial information of C^* and T have some similarity in semi-online algorithm design and analysis, especially for the strongly related problem of minimizing the maximal machine load over identical machines [3,7]. However, the situation is quite different for machine covering. For $Pm|opt|C_{\min}$, the competitive ratio of the algorithm $FILL$ converges to 2 when $m \to \infty$, while the lower bound is even smaller [1]. For $Pm|sum|C_{\min}$, the competitive ratio of the algorithm, which matches the lower bound, tends to infinity when $m \to \infty$. There are still no constant competitive ratio algorithms for the problem where both T and $p_{\max}$ are known in advance. These results indicated that C^* is much more useful than T for designing semi-online algorithms for machine covering.

This paper is organized as follows. Sections 2 and 3 propose lower bounds and optimal algorithms for the semi-online problems $Pm|sum|C_{\min}$ and $Pm|max|C_{\min}$, respectively. Section 4 is devoted to the problem $Pm|sum\ \&\ max|C_{\min}$.

In this paper, when analyzing a semi-online algorithm A, we denote by l_i^t the current load of M_i right before the assignment of p_t, and by L_i the load of M_i after all the jobs have been assigned, $i = 1, \ldots, m$. Thus $C^A = \min_{1 \leq i \leq m} L_i$. We denote by p_B the first job with processing time $p_{\max}$.

2. Optimal algorithm for $Pm|sum|C_{\min}$

Theorem 2.1. *The competitive ratio of any semi-online algorithm A for the problem $Pm|sum|C_{\min}$, $m \geq 3$, is at least $m - 1$.*

Proof. The proof will be proved by an adversary argument. Let $T = m$. The first $m-1$ jobs all have the same processing time $\frac{1}{m-1}$. If an algorithm A assigns them to $m-1$ different machines, then $m-1$ jobs with the same processing time 1 arrive. We have $C^A \le \frac{1}{m-1}$ while $C^* = 1$. It follows that $\frac{C^*}{C^A} \ge m-1$. If the first $m-1$ jobs are assigned to less than $m-1$ machines by algorithm A, the next and last job with processing time $m-1$ arrives. We have $C^A = 0$ while $C^* = \frac{1}{m-1}$. It also follows that $\frac{C^*}{C^A} \ge m-1$. Therefore, we conclude that for any semi-online algorithm A, the competitive ratio of A is at least $m-1$. □

In the rest of this section, we present a semi-online Algorithm $H1$ and prove it is the optimal one.

Algorithm $H1$

1. Let $\mathcal{M}_1 = \{M_1, M_2, \ldots, M_{m-1}\}$. $s = 1$.
2. Always assign current job to one machine in $\mathcal{M}_s$ chosen by the LS rule until there exists a job p_{h_s}, such that assigning p_{h_s} to any machine will result in the load of this machine being at least $\frac{T}{2m}$, i.e.

$$l_i^{h_s} < \frac{T}{2m} \quad \text{and} \quad l_i^{h_s} + p_{h_s} \ge \frac{T}{2m}, \ \forall M_i \in \mathcal{M}_s.$$

Let M_{i_s} be the machine with the smallest current load in $\mathcal{M}_s$ right before the assignment of p_{h_s}, i.e.

$$l_{i_s}^{h_s} = \min_{M_i \in \mathcal{M}_s} l_i^{h_s}. \tag{1}$$

(2.1) If $l_{i_s}^{h_s} + p_{h_s} > \frac{T}{m}$, assign p_{h_s} to M_m. Then assign all remaining jobs to $\mathcal{M}_s$ by LS rule. Stop.

(2.2) If $l_{i_s}^{h_s} + p_{h_s} \le \frac{T}{m}$, assign p_{h_s} to M_{i_s} and goto Step 3.

3. If $s < m-1$, let $\mathcal{M}_{s+1} = \mathcal{M}_s \backslash \{M_{i_s}\}$, $s = s+1$, goto Step 2. Otherwise, assign all remaining jobs to M_m. Stop.

The main idea of $H1$ is as follows. Since $C^* \le \frac{T}{m}$, it is sufficient to prove $C^{H1} = \min_{1 \le i \le m} L_i \ge \frac{T}{m(m-1)}$ guarantees the competitive ratio of $H1$ to be no greater than $m-1$. Therefore, there is no necessity to process any more jobs for machines with current loads greater than $\frac{T}{2m} \ge \frac{T}{m(m-1)}$. On the other hand, we must avoid the load of some machine becoming too large. Hence we leave machine M_m unused from the beginning until there exists a large job which will cause the load of some machine to be greater than $\frac{T}{m}$, or other machines all with loads greater than $\frac{T}{2m}$. For the former case, assigning the large job to M_m will make its load greater than $\frac{T}{2m}$. For the latter case, only the current load of M_m is less than $\frac{T}{2m}$, therefore we assign all remaining jobs to M_m. The following lemma generalizes the upper bound of $C^* \le \frac{T}{m}$ and is useful throughout the paper.

Lemma 2.1 (*[10]*). *For each* $1 \le s \le m-1$, $C^* \le \frac{T - \sum_{k=1}^{s} p_{j_k}}{m-s}$, *where* $p_{j_1}, p_{j_2}, \ldots, p_{j_{m-1}}$ *are arbitrary* $m-1$ *jobs in* $\mathcal{J}$.

Proof. Assume that there exists some s, $1 \le s \le m-1$, such that $C^* > \frac{T - \sum_{k=1}^{s} p_{j_k}}{m-s}$. It is clear that there exists at least $m-s$ machines not processing any jobs in $\{p_{j_1}, p_{j_2}, \ldots, p_{j_s}\}$ in the optimal schedule. From the assumption, we know that the load of any of these $m-s$ machines is greater than $\frac{T - \sum_{k=1}^{s} p_{j_k}}{m-s}$. Hence the total load of all m machines is greater than $(m-s)\frac{T - \sum_{k=1}^{s} p_{j_k}}{m-s} + \sum_{k=1}^{s} p_{j_k} = T$, which is a contradiction. □

Theorem 2.2. *The competitive ratio of the algorithm* $H1$ *for the problem* $Pm|sum|C_{\min}$, $m \ge 3$, *is at most* $m-1$.

Proof. W.l.o.g., we normalize the processing time of all jobs in such a way that $T = m$. Hence, $C^* \le 1$. We distinguish two cases according to the value of s when $H1$ terminates.

Case 1 $s = m-1$.

In this case, we have $\frac{1}{2} \le l_{i_k}^{h_k} + p_{h_k} \le 1$ and p_{h_k} is assigned to M_{i_k}, $k = 1, \ldots, m-1$ (see Fig. 1(a)). Note that $(i_1\ i_2\ \cdots\ i_{m-1})$ is a permutation of $(1\ 2\ \cdots\ m-1)$. Since p_{h_k} is the last job assigned to M_{i_k}, $k = 1, \ldots, m-1$, and all jobs after $p_{h_{m-1}}$ are assigned to M_m, we have

$$\frac{1}{2} \le L_{i_k} = l_{i_k}^{h_k} + p_{h_k} \le 1, \quad 1 \le k \le m-1,$$

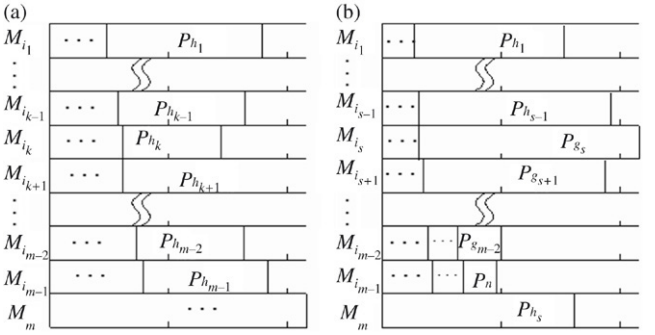

Fig. 1. (a) Case 1: $s = m-1$. (b) Case 2: $s < m-1$.

$$L_m = T - \sum_{k=1}^{m-1} L_{i_k} = T - \sum_{k=1}^{m-1} (l_{i_k}^{h_k} + p_{h_k}) \geq m - (m-1) > \frac{1}{2},$$

$$C^{H1} = \min_{1\leq i\leq m} L_i = \min\{\min_{1\leq k\leq m-1} L_{i_k}, L_m\} \geq \frac{1}{2} \geq \frac{1}{m-1} \geq \frac{1}{m-1} C^*.$$

Case 2 $s < m-1$.

In this case, loads of M_{i_k}, $1 \leq k \leq s-1$, lie in the interval $[\frac{1}{2}, 1]$ right before the assignment of p_{h_s}, and these machines will not process any more jobs. Hence, we have

$$L_{i_k} = l_{i_k}^{h_k} + p_{h_k} \geq \frac{1}{2}, \quad k = 1, \ldots, s-1. \tag{2}$$

Moreover, since $l_{i_s}^{h_s} < \frac{1}{2}$, $l_{i_s}^{h_s} + p_{h_s} > 1$ and p_{h_s} is assigned to M_m, we have

$$L_m = p_{h_s} > \frac{1}{2}. \tag{3}$$

Denote the remaining $m-s$ machines in $\mathcal{M}_s = \mathcal{M}\backslash\{M_{i_1}, \ldots, M_{i_{s-1}}, M_m\}$ by $M_{i_s}, \ldots, M_{i_{m-1}}$ in such an order that $L_{i_s} \geq L_{i_{s+1}} \geq \cdots \geq L_{i_{m-1}}$. Note that $(i_1 \ \cdots \ i_{m-1})$ is still a permutation of $(1 \ \cdots \ m-1)$. If $L_{i_{m-1}} \geq \frac{1}{m-1}$, combining with (2) and (3), we have

$$C^{H1} = \min\left\{\min_{1\leq k\leq s-1} L_{i_k}, \min_{s\leq k\leq m-1} L_{i_k}, L_m\right\} \geq \frac{1}{m-1} \geq \frac{1}{m-1} C^*.$$

Otherwise, $C^{H1} = L_{i_{m-1}} < \frac{1}{m-1}$. Denote by p_{g_k} the last job assigned to M_{i_k}, $s \leq k \leq m-2$ (see Fig. 1(b)). Since p_{g_k} is assigned by the LS rule, we have

$$L_{i_k} - p_{g_k} = l_{i_k}^{g_k} \leq l_{i_{m-1}}^{g_k} \leq L_{i_{m-1}}, \quad s \leq k \leq m-2. \tag{4}$$

On the other hand, by (1), we have

$$l_{i_k}^{h_k} = \min_{M_i\in\mathcal{M}_k} l_i^{h_k} \leq l_{i_{m-1}}^{h_k} \leq L_{i_{m-1}}, \quad 1 \leq k \leq s-1. \tag{5}$$

Since $p_{h_1}, p_{h_2}, \ldots, p_{h_{s-1}}, p_{h_s}, p_{g_s}, \cdots, p_{g_{m-2}}$ are $m-1$ jobs in $\mathcal{J}$, by Lemma 2.1 and (2)–(5), we obtain

$$\begin{aligned} C^* &\leq \frac{T - \left(\sum_{k=1}^{s-1} p_{h_k} + \sum_{k=s}^{m-2} p_{g_k} + p_{h_s}\right)}{m-(m-1)} = \sum_{i=1}^{m} L_i - \left(\sum_{k=1}^{s-1} p_{h_k} + \sum_{k=s}^{m-2} p_{g_k} + p_{h_s}\right) \\ &= \sum_{k=1}^{s-1} \left(L_{i_k} - p_{h_k}\right) + \sum_{k=s}^{m-2} \left(L_{i_k} - p_{g_k}\right) + L_{i_{m-1}} + (L_m - p_{h_s}) \\ &= \sum_{k=1}^{s-1} l_{i_k}^{h_k} + \sum_{k=s}^{m-2} l_{i_k}^{g_k} + L_{i_{m-1}} + 0 \leq (s-1)L_{i_{m-1}} + (m-s-1)L_{i_{m-1}} + L_{i_{m-1}} \\ &= (m-1)L_{i_{m-1}} = (m-1)C^{H1}. \quad \square \end{aligned}$$

From Theorems 2.1 and 2.2, we know that $H1$ is the optimal algorithm for $Pm|sum|C_{\min}$, $m \geq 3$, with competitive ratio $m-1$.

3. Optimal algorithm for $Pm|max|C_{\min}$

Theorem 3.1. *The competitive ratio of any semi-online algorithm A for the problem $Pm|max|C_{\min}$, $m \geq 3$, is at least $m-1$.*

Proof. Let $p_{\max} = m-1$ be known in advance. The first $m-1$ jobs all have the same processing time 1. If an algorithm A assigns them to $m-1$ different machines, then $m-1$ jobs with the same processing time $m-1$ arrive. We have $C^A \leq 1$ while $C^* = m-1$. It follows that $\frac{C^*}{C^A} \geq m-1$. If the first $m-1$ jobs are assigned to less than $m-1$ machines by algorithm A, then the next and last job with the processing time $m-1$ arrives. We have $C^A = 0$ while $C^* = 1$. It follows that $\frac{C^*}{C^A} \geq m-1$. Therefore, we conclude that for any semi-online algorithm A, the competitive ratio of A is at least $m-1$. □

Note that instances used in the proof of Theorem 3.1 are essentially the same as those in the proof of Theorem 2.1. The following algorithm $H2$ is modified from PLS [11], and also has some similarity with $H1$. $H2$ consists of two stages. We keep M_m unused during the former stage in order to leave some room for p_B. On the other hand, avoid too large a difference between the loads of machines. In the latter stage, we simply assign jobs by the LS rule.

Algorithm $H2$

1. Always assign current job to one machine in $\mathcal{M}\backslash\{M_m\}$ chosen by LS rule until one of the following cases happens.
   (1.1) The current job is p_B.
   (1.2) If the current job is assigned to the machine chosen by LS rule, the new load of this machine would be greater than $2p_{\max}$.
2. Once (1.1) or (1.2) happens, then assign the current job to M_m. Thereafter assign all the remaining jobs by LS rule. Stop.

Before giving the proof of the competitive ratio of $H2$, we give two lemmas about the loads of machines after all jobs have been assigned by $H2$.

Lemma 3.1. $L_m \geq p_{\max}$.

Proof. If p_B is the first job assigned to M_m, then it is clear that $L_m \geq p_{\max}$. Otherwise, assume p_a to be the first job assigned to M_m, which comes before p_B. Then we have $l_i^a + p_a > 2p_{\max}$, $1 \leq i \leq m-1$, i.e.

$$l_i^a > 2p_{\max} - p_a > p_{\max}, \quad 1 \leq i \leq m-1. \tag{6}$$

By Step 2 of $H2$, all remaining jobs including p_B are assigned by LS rule. If p_B is assigned to M_m, the lemma is obviously true. If p_B is assigned to M_i, $i < m$, it implies that $l_i^B \leq l_m^B$. Therefore, $L_m \geq l_m^B \geq l_i^B \geq l_i^a > p_{\max}$ and we are done. □

Lemma 3.2. $|L_i - L_k| \leq p_{\max}$, $1 \leq i, k \leq m$.

Proof. Suppose M_i and M_k be any two machines with $L_i \leq L_k$, $1 \leq i, k \leq m-1$. Let p_{j_k} be the last job assigned to M_k. Since p_{j_k} is assigned by LS rule and M_i, M_k are both candidate machines. We have $L_k - p_{j_k} = l_k^{j_k} \leq l_i^{j_k} \leq L_i$ and thus $|L_i - L_k| \leq p_{j_k} \leq p_{\max}$.

Next, we prove $|L_i - L_m| \leq p_{\max}$, $1 \leq i \leq m-1$, by contradiction. If for some i, $1 \leq i \leq m-1$, $L_i - L_m < -p_{\max}$. Denoted by p_{j_m} the last job assigned to M_m, then

$$l_m^{j_m} = L_m - p_{j_m} \geq L_m - p_{\max} > L_i \geq l_i^{j_m}.$$

This implies that p_{j_m} should be assigned to M_i, which violates the definition of p_{j_m}. If for some i, $1 \leq i \leq m-1$, $L_i - L_m > p_{\max}$, let p_{j_i} be the last job assigned to M_i. By Lemma 3.1, $l_i^{j_i} + p_{j_i} = L_i > L_m + p_{\max} \geq 2p_{\max}$, so p_{j_i} should be assigned to the machine in $\mathcal{M}$ with the smallest current load. However, $l_i^{j_i} = L_i - p_{j_i} \geq L_i - p_{\max} > L_m \geq l_m^{j_i}$, which is a contradiction. □

Theorem 3.2. *The competitive ratio of the algorithm $H2$ for $Pm|max|C_{\min}$, $m \geq 3$, is at most $m-1$.*

Proof. We distinguish two cases according to the first job assigned to M_m.

Case 1 The first job assigned to M_m is p_B.

By Lemma 3.2, we have $|L_i - L_k| \leq p_{\max}$, $1 \leq i, k \leq m$. Therefore,

$$L_i \leq \min_{1\leq k\leq m} L_k + p_{\max} = C^{H2} + p_{\max}, \quad 1 \leq i \leq m,$$

$$C^* \leq \frac{1}{m}T = \frac{1}{m}\sum_{i=1}^{m} L_i \leq \frac{1}{m}\left(C^{H2} + (m-1)(C^{H2} + p_{\max})\right) = C^{H2} + \frac{m-1}{m}p_{\max}.$$

If $C^{H2} \geq p_{\max}$, then

$$C^* \leq C^{H2} + \frac{m-1}{m}p_{\max} \leq \frac{2m-1}{m}C^{H2} \leq (m-1)C^{H2}, \tag{7}$$

where the last inequality is valid when $m \geq 3$. Otherwise, $C^{H2} < p_{\max} \leq L_m$. W.l.o.g., we assume that $C^{H2} = L_{m-1}$. Note that for this case,

$$L_m = p_{\max}. \tag{8}$$

In fact, p_B is the first job assigned to M_m. If there exists a job p_d assigned to M_m after p_B, then $l^d_{m-1} \leq L_{m-1} < p_{\max} \leq l^d_m$, which contradicts to the fact that p_d is assigned by LS rule.

Denote by p_{j_i} the last job assigned to M_i, $1 \leq i \leq m-2$. Since they are assigned by LS rule, we have

$$L_i - p_{j_i} = l_i^{j_i} \leq l_{m-1}^{j_i} \leq L_{m-1}, \quad 1 \leq i \leq m-2. \tag{9}$$

By Lemma 2.1 and (8) and (9), we obtain

$$\begin{aligned} C^* &\leq \frac{1}{m-(m-1)}\left(T - \left(\sum_{i=1}^{m-2} p_{j_i} + p_B\right)\right) = \sum_{i=1}^{m} L_i - \sum_{i=1}^{m-2} p_{j_i} - p_{\max} \\ &= \sum_{i=1}^{m-2}(L_i - p_{j_i}) + L_{m-1} + (L_m - p_{\max}) \\ &\leq (m-2)L_{m-1} + L_{m-1} = (m-1)C^{H2}. \end{aligned}$$

Case 2 The first job assigned to M_m is $p_a < p_B$.

By (6) and Lemma 3.1, we have $L_i \geq l_i^a > p_{\max}$, $1 \leq i \leq m-1$ and $L_m \geq p_{max}$. Therefore, $C^{H2} \geq p_{\max}$, (7) can finish the proof. □

From Theorems 3.1 and 3.2, we know that $H2$ is the optimal algorithm for $Pm|max|C_{\min}$, $m \geq 3$, with competitive ratio $m-1$.

4. Optimal algorithms for $Pm|sum\ \&\ max|C_{\min}$

Theorem 4.1. *The competitive ratio of any semi-online algorithm A for the problem $P3|sum\ \&\ max|C_{\min}$ is at least $\frac{3}{2}$.*

Proof. Let $T = 9$ and $p_{\max} = 3$. The first two jobs are $p_1 = p_2 = 1$. If an algorithm A assigns them to the same machine, let the last three jobs be $p_3 = p_4 = 2$ and $p_5 = 3$. Then $C^A \leq 2$ while $C^* = 3$. It follows that $\frac{C^*}{C^A} \geq \frac{3}{2}$. If the first two jobs are assigned to different machines by algorithm A, let the last three jobs be $p_3 = 1$ and $p_4 = p_5 = 3$. We also have $C^A \leq 2$ while $C^* = 3$, thus $\frac{C^*}{C^A} \geq \frac{3}{2}$. Therefore, we conclude that for any semi-online algorithm A, the competitive ratio of A is at least $\frac{3}{2}$. □

Now we are going to propose an algorithm $H3$. The design of the algorithm depends on the ratio between T and $p_{\max}$. When $T/p_{\max}$ is too big, all jobs have similar processing times, and hence the LS rule works. For the remaining cases, we assume that p_B is the first job of the sequence and we always assign it to M_3. Such an assumption will not affect strictness since the partial information ensures the existence of p_B with known processing time $p_{\max}$. One can easily modify the Step 2 and 3 of the algorithm $H3$ given below to solve instances without the above assumption by always taking p_B into account when considering the load of M_3. In other words, modify the definition of the current load of M_3 right before the assignment of p_j to be

$$\bar{l}_3^j = \begin{cases} l_3^j + p_{\max} & \text{if } p_j \text{ comes before } p_B, \\ l_3^j & \text{otherwise.} \end{cases}$$

Algorithm $H3$

1. **For** $0 < p_{\max} \le \frac{T}{6}$, assign all jobs by LS rule.
2. **For** $\frac{2T}{9} \le p_{\max} \le T$.
   (2.1) Assign p_B to M_3.
   (2.2) Assign jobs to M_1 until there exists a job p_f, such that $l_1^f < \frac{1}{3}(T - p_{\max})$ and $l_1^f + p_f \ge \frac{1}{3}(T - p_{\max})$.
      (2.2.1) If $l_1^f + p_f \le \frac{2}{3}(T - p_{\max})$, assign p_f to M_1 and all remaining jobs to M_2. Stop.
      (2.2.2) If $l_1^f + p_f > \frac{2}{3}(T - p_{\max})$, assign p_f to M_2 and all remaining jobs to M_1. Stop.
3. **For** $\frac{T}{6} < p_{\max} < \frac{2T}{9}$.
   (3.1) Assign p_B to M_3.
   (3.2) **While** there exists at least one unassigned job. Suppose p_j is the current job, define
   $$U^j = \left\{ M_i | l_i^j < \frac{2T}{9}, l_i^j + p_j \le \frac{T}{3},\ i \in \{1, 2\} \right\},\ V^j = \{M_i | l_i^j \le p_{\max},\ i \in \{1, 2\}\}.$$
      (3.2.1) If $M_1 \in U^j$, assign p_j to M_1.
      (3.2.2) If $M_1 \notin U^j$ and $M_2 \in U^j$, assign p_j to M_2.
      (3.2.3) If $U^j = \emptyset$ and $M_2 \in V^j$, assign p_j to M_2.
      (3.2.4) If $U^j = \emptyset$, $M_2 \notin V^j$ and $M_1 \in V^j$, assign p_j to M_1.
      (3.2.5) If $U^j = V^j = \emptyset$, assign p_j by LS rule.

From the description of Step 3.2, we can conclude that if p_j is assigned to M_3, then we must have $U^j = V^j = \emptyset$ and $l_3^j \le l_i^j,\ i = 1, 2$. If $M_i \notin U^j \cup V^j$ and p_j is still assigned to M_i then $l_i^j = \min\{l_1^j, l_2^j, l_3^j\},\ i = 1, 2$.

Theorem 4.2. *The competitive ratio of the algorithm $H3$ for $P3|sum\ \&\ max|C_{\min}$ is at most $\frac{3}{2}$.*

Proof. For $0 < p_{\max} \le \frac{T}{6}$, since jobs are assigned by LS rule, we have

$$\begin{aligned} &|L_i - L_k| \le p_{\max}, \quad 1 \le i, k \le 3, \\ &L_i \le C^{H3} + p_{\max}, \quad i = 1, 2, 3, \\ &C^* \le \frac{T}{3} = \frac{L_1 + L_2 + L_3}{3} \le \frac{2(C^{H3} + p_{\max}) + C^{H3}}{3} = C^{H3} + \frac{2}{3}p_{\max}, \\ &\frac{C^*}{C^{H3}} \le \frac{\frac{T}{3}}{\frac{T}{3} - \frac{2}{3}p_{\max}} \le \frac{\frac{T}{3}}{\frac{T}{3} - \frac{T}{9}} = \frac{3}{2}. \end{aligned}$$

For $\frac{2T}{9} \le p_{\max} \le T$, it is clear that $L_3 = p_{\max}$ and $C^* \le \frac{T}{3} \le \frac{3}{2}p_{\max} = \frac{3}{2}L_3$. Therefore, in order to prove $C^* \le \frac{3}{2}C^{H3} = \frac{3}{2}\min\{L_1, L_2, L_3\}$, we only need to show $C^* \le \frac{3}{2}L_1$ and $C^* \le \frac{3}{2}L_2$. By Lemma 2.1, we have $C^* \le \frac{T - p_{\max}}{3-1}$, and thus it is sufficient to prove $L_1 \ge \frac{1}{3}(T - p_{\max})$ and $L_2 \ge \frac{1}{3}(T - p_{\max})$. We distinguish two cases according to the value of $l_1^f + p_f$.

Case 1 $l_1^f + p_f \le \frac{2}{3}(T - p_{\max})$.

By the definition of p_f, we have $l_1^f + p_f \geq \frac{1}{3}(T - p_{\max})$. Therefore,

$$L_1 = l_1^f + p_f \geq \frac{1}{3}(T - p_{\max}), \quad L_2 = T - L_3 - L_1 = T - p_{\max} - (l_1^f + p_f) \geq \frac{1}{3}(T - p_{\max}).$$

Case 2 $l_1^f + p_f > \frac{2}{3}(T - p_{\max})$.

By Step 2.2.2, only p_f is assigned to M_2. Since $l_1^f < \frac{1}{3}(T - p_{\max})$, we have $p_f > \frac{1}{3}(T - p_{\max})$. Therefore, $L_2 = p_f \geq \frac{1}{3}(T - p_{\max})$. On the other hand,

$$L_1 = T - L_3 - L_2 = T - p_{\max} - p_f = \frac{T - (p_{\max} + p_f)}{3 - 2} \geq C^*,$$

the last inequality is also due to Lemma 2.1.

The case of $\frac{2T}{9} \leq p_{\max} \leq T$ is thus finished.

In the rest of the proof, we concentrate on the case of $\frac{T}{6} < p_{\max} < \frac{2T}{9}$. We first give some lemmas about the loads of three machines after all jobs have been assigned.

Lemma 4.1. *If* $L_{i_1} < \frac{2T}{9}$, *then* $L_{i_2} > \frac{T}{3}$ *and* $L_{i_3} > \frac{T}{3}$, *where* $(i_1\ i_2\ i_3)$ *is any permutation of* (1 2 3).

Proof. Suppose $L_{i_2} \leq \frac{T}{3}$, then $L_{i_3} = T - L_{i_1} - L_{i_2} > T - \frac{2T}{9} - \frac{T}{3} = \frac{4T}{9}$. Denote by p_e the last job assigned to M_{i_3}. Because $L_{i_3} > \frac{4T}{9} > p_{\max}$ and we assume p_B is the first job of the sequence, p_e can not be p_B. Therefore, we have $l_{i_3}^e = L_{i_3} - p_e \geq L_{i_3} - p_{\max} > \frac{2T}{9} > p_{\max}$. By the definition of U^e and V^e, we know that $M_{i_3} \notin U^e \cup V^e$. It follows that p_e will not be assigned to M_{i_3} unless $l_{i_3}^e \leq l_{i_1}^e$ and $l_{i_3}^e \leq l_{i_2}^e$, which contradicts $l_{i_3}^e > \frac{2T}{9} > L_{i_1} \geq l_{i_1}^e$. $L_{i_2} > \frac{T}{3}$ is thus proved. $L_{i_3} > \frac{T}{3}$ can be proved in the same way. □

Lemma 4.2. *If* $L_1 < \frac{2T}{9}$ *or* $L_2 < \frac{2T}{9}$, *then* $L_i > p_{\max},\ i = 1, 2, 3$.

Proof. If $L_1 < \frac{2T}{9}$, by Lemma 4.1, we have

$$L_3 > \frac{T}{3} > p_{\max}. \tag{10}$$

Hence, there exist some jobs assigned to M_3 besides p_B. Denote one of such jobs by p_r. Since p_r will not be assigned to M_3 unless $U^r = V^r = \emptyset$, we have $L_i \geq l_i^r > p_{\max},\ i = 1, 2$. Together with (10), the lemma is thus proved. If $L_2 < \frac{2}{9}T$, the result can be proved similarly. □

Now we are going to prove $C^* \leq \frac{3}{2}C^{H3}$. Obviously, if $C^{H3} \geq \frac{2T}{9}$, then $C^* \leq \frac{T}{3} = \frac{3}{2} \cdot \frac{2T}{9} \leq \frac{3}{2}C^{H3}$. Therefore, we assume $C^{H3} < \frac{2T}{9}$ in the following. We further distinguish three cases to reach the final result. Denote by p_x, p_y the last jobs assigned to M_1, M_2, respectively.

Case 1 $C^{H3} = L_3 < \frac{2T}{9}$.

Obviously,

$$L_3 \geq p_{\max} \tag{11}$$

since p_B is always assigned to M_3. Moreover, by Lemma 4.1, we have

$$L_1 = l_1^x + p_x > \frac{T}{3}, \qquad L_2 = l_2^y + p_y > \frac{T}{3}. \tag{12}$$

We first show that

$$L_1 - p_x = l_1^x \leq L_3 < \frac{2T}{9}. \tag{13}$$

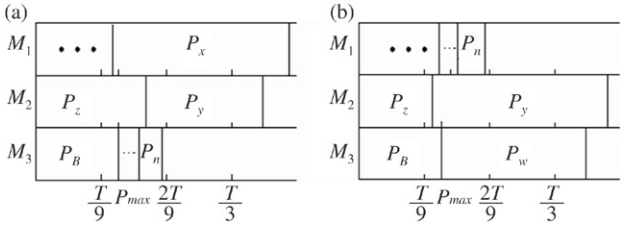

Fig. 2. (a) Subcase 1: $C^{H3} = L_3 < \frac{2T}{9}$. (b) Subcase 3: $C^{H3} = L_1 < \frac{2T}{9}$.

By (12), $M_1 \notin U^x$. If $M_1 \in V^x$, by the definition of V^x and (11), $l_1^x \le p_{\max} \le L_3$ and we are done. If $M_1 \notin V^x$, since p_x is assigned to M_1, we have $l_1^x \le l_3^x \le L_3$. (13) is thus proved,

$$L_2 - p_y = l_2^y \le L_3 < \frac{2T}{9}. \tag{14}$$

can be proved similarly.

Next, we will show that there is only one job assigned to M_2 before p_y. Suppose that there are at least two jobs assigned to M_2 before the assignment of p_y with total processing time $l_2^y < \frac{2T}{9}$, thus at least one job with processing time less than $\frac{T}{9}$ exists. Let $p_q < \frac{T}{9}$ be the first such job.

Subcase 1 p_x comes before p_q.

We have already proved $M_1 \notin U^x$. Since p_x is not assigned to M_2, $M_2 \notin U^x$, $M_2 \notin V^x$ and thus $l_2^x > p_{\max}$. As p_y is the last job assigned to M_2, p_q comes before p_y. Also p_x comes before p_q, and we have $l_2^y \ge l_2^q \ge l_2^x > p_{\max}$. Hence, $M_2 \notin V^y$. Note that we also have $M_2 \notin U^y$ by (12). So p_y is assigned to M_2 by LS rule and $l_3^y \ge l_2^y > p_{\max}$, which implies that there already exists some job, denoted by p_t, assigned to M_3 before the assignment of p_y. But this also causes a contradiction. In fact, as $p_t \le L_3 - p_B < \frac{2}{9}T - p_{\max} < \frac{1}{18}T$, together with (14), we have $l_2^t \le l_2^y < \frac{2T}{9}$ and $l_2^t + p_t < \frac{2T}{9} + \frac{T}{18} < \frac{T}{3}$. It follows that $M_2 \in U^t$ and p_t should be assigned to M_2.

Subcase 2 p_q comes before p_x.

In this case, we have $l_1^q \le l_1^x \le L_3 < \frac{2T}{9}$ and $l_1^q + p_q \le \frac{2T}{9} + \frac{T}{9} = \frac{T}{3}$. It follows that $M_1 \in U^q$ and p_q should be assigned to M_1, also a contradiction.

Combining with the above discussion, we know that there are only two jobs assigned to M_2, one is p_y and we denote the other one by p_z(see Fig. 2(a)), thus

$$L_2 = p_z + p_y. \tag{15}$$

Obviously, at least two of the four jobs p_x, p_y, p_z and p_B must be assigned to the same machine in the optimal schedule. Assume that p_{j_1} and p_{j_2}, $j_1, j_2 \in \{x, y, z, B\}$ are those two jobs. Similar to the proof of Lemma 2.1, we can prove $C^* \le (T - (p_{j_1} + p_{j_2}))/2$. By (11), (13) and (15), we obtain

$$\begin{aligned} C^* &\le \frac{T - (p_{j_1} + p_{j_2}) + (4p_{\max} - p_x - p_y - p_z - p_B)}{2} \\ &\le \frac{T + 2p_{\max} - (p_x + p_y + p_z + p_B)}{2} \\ &= \frac{(L_1 - p_x) + (L_2 - p_z - p_y) + L_3 - p_B + 2p_{\max}}{2} \\ &\le \frac{L_3 + 0 + L_3 + p_{\max}}{2} \le \frac{3}{2}L_3 = \frac{3}{2}C^{H3}. \end{aligned}$$

Case 2 $C^{H3} = L_2 < \frac{2T}{9}$.

We show that this case is impossible. Since $C^{H3} = L_2 < \frac{2T}{9}$, by Lemma 4.2, we have $L_2 > p_{\max}$. Therefore, there are at least two jobs assigned to M_2 with total processing time less than $\frac{2T}{9}$, thus at least one job with processing time

less than $\frac{T}{9}$ exists. Let $p_q < \frac{T}{9}$ be the first such job. By Lemma 4.1, we have $l_1^x + p_x = L_1 > \frac{T}{3}$ and thus $M_1 \notin U^x$. Since p_x is assigned to M_1, we have

$$M_2 \notin U^x \cup V^x \tag{16}$$

and

$$l_1^x \le l_2^x \le L_2 < \frac{2T}{9}. \tag{17}$$

Subcase 1 p_x comes before p_q.

By (16), we have $l_2^x > p_{\max}$. Together with (17), we know that there are at least two jobs assigned to M_2 with total processing time less than $\frac{2T}{9}$ before the assignment of p_x. Hence, one job with processing time less than $\frac{T}{9}$ is assigned to M_2 before the assignment of p_q, which contradicts the definition of p_q.

Subcase 2 p_q comes before p_x.

By (17), we have $l_1^q \le l_1^x < \frac{2T}{9}$ and $l_1^q + p_q \le l_1^x + p_q \le \frac{2T}{9} + \frac{T}{9} = \frac{T}{3}$. Hence, $M_1 \in U^q$ and p_q should be assigned to M_1, which is a contradiction.

From the above discussion, we know that $C^{H3} = L_2 < \frac{2T}{9}$ is impossible.

Case 3 $C^{H3} = L_1 < \frac{2T}{9}$.

Since the current load of M_1 is always less than $\frac{2T}{9}$, the processing time of the jobs assigned to M_2 and M_3 must be greater than $\frac{T}{9}$, otherwise, they will be assigned to M_1 by Step 3.2.1. Moreover, there are only two jobs assigned to M_2. In fact, suppose p_s is the third job assigned to M_2, then $l_2^s > 2 \times \frac{T}{9} > p_{\max}$ and thus $M_2 \notin U^s \cup V^s$. Since $l_2^s > \frac{2T}{9} > L_1 \ge l_1^s$, p_s can not be assigned to M_2 even by LS rule, which is a contradiction. Denote the job assigned to M_2 other than p_y by p_z(see Fig. 2(b)) and we have

$$L_2 = p_z + p_y. \tag{18}$$

Furthermore, there is only one job, denoted by p_w, assigned to M_3 besides p_B. In fact, if p_t is the third job assigned to M_3, then $l_3^t > p_w + p_B > \frac{T}{9} + p_{\max} > \frac{2T}{9} > L_1 \ge l_1^t$, which implies that p_t can not be assigned to M_3 by LS rule. Hence,

$$L_3 = p_w + p_B. \tag{19}$$

Similarly to the Case 1, at least two of p_x, p_y, p_z and p_B must be assigned to the same machine in the optimal schedule. Combining with (18) and (19) and $L_1 > p_{\max}$, which is due to Lemma 4.2, we have

$$\begin{aligned} C^* &\le \frac{T + 2p_{\max} - (p_y + p_z + p_w + p_B)}{2} \\ &= \frac{L_1 + (L_2 - p_z - p_y) + (L_3 - p_w - p_B) + 2p_{\max}}{2} \\ &= \frac{L_1 + 2p_{\max}}{2} \le \frac{3L_1}{2} = \frac{3}{2}C^{H3}. \quad \square \end{aligned}$$

From Theorems 4.1 and 4.2, we know that $H3$ is the optimal algorithm for $P3|sum\ \&\ max|C_{\min}$ with competitive ratio $\frac{3}{2}$.

Theorem 4.3. *The competitive ratio of any semi-online algorithm A for the problem $Pm|sum\ \&\ max\ |C_{\min}$, $m \ge 4$, is at least $m - 2$.*

Proof. Let $T = 2m - 3$ and $p_{\max} = m - 2$. The first $m - 2$ jobs have the same processing time $\frac{1}{m-2}$. If an algorithm A assigns them to $m - 2$ different machines, then $m - 2$ jobs with the same processing time 1 and the last job with processing time $m - 2$ arrive. We have $C^A \le \frac{1}{m-2}$ while $C^* = 1$. It follows that $\frac{C^*}{C^A} \ge m - 2$. If the first $m - 2$ jobs are assigned to less than $m - 2$ machines by algorithm A, then the last two jobs with the same processing time $m - 2$ arrive. It follows that $C^A = 0$ while $C^* = \frac{1}{m-2}$, then $\frac{C^*}{C^A} > m - 2$. Therefore, we conclude that for any semi-online algorithm A, the competitive ratio of A is at least $m - 2$. $\square$

In the rest of the paper, we will present an optimal algorithm for $m \geq 4$ machines. When $T/p_{\max}$ is large enough, the LS rule still works. Otherwise, we assign p_B to M_m solely. To assign the remaining jobs in $\mathcal{J}' = \mathcal{J}\backslash\{p_B\}$ to machine set $\mathcal{M}' = \mathcal{M}\backslash\{M_m\}$, as $(\mathcal{M}', \mathcal{J}')$ is an instance of $Pm'|sum|C_{\min}$ with $m' = m-1$ and known total processing time $T' = T - p_{\max}$, we use algorithm $H1$ as a procedure. According to the competitive ratio of $H1$ and by revealing the relations between $C^{H4}(\mathcal{M}, \mathcal{J})$ and $C^{H1}(\mathcal{M}', \mathcal{J}')$, $C^*(\mathcal{M}, \mathcal{J})$ and $C^*(\mathcal{M}', \mathcal{J}')$, the competitive ratio of $H4$ can be proved.

Algorithm $H4$

1. **For** $0 < p_{\max} \leq \frac{(m-3)}{(m-1)(m-2)}T$, assign all jobs by LS rule.
2. **For** $\frac{(m-3)}{(m-1)(m-2)}T < p_{\max} \leq T$,
   2.1 Assign p_B to M_m whenever it arrives.
   2.2 Assign $\mathcal{J}'$ to $\mathcal{M}'$ by $H1$.

Theorem 4.4. *The competitive ratio of algorithm $H4$ for the problem $Pm|sum\ \&\ max|C_{\min}$, $m \geq 4$ is at most $m-2$.*

Proof. For $0 < p_{\max} \leq \frac{(m-3)}{(m-1)(m-2)}T$, since jobs are assigned by LS rule, we have

$$|L_i - L_k| \leq p_{\max}, \quad 1 \leq i, k \leq m,$$

$$C^* \leq \frac{T}{m} = \frac{\sum_{i=1}^{m} L_i}{m} \leq \frac{(m-1)(C^{H4} + p_{\max}) + C^{H4}}{m} \leq C^{H4} + \frac{m-1}{m}p_{\max},$$

$$\frac{C^*}{C^{H4}} \leq \frac{C^{H4} + \frac{m-1}{m}p_{\max}}{C^{H4}} \leq 1 + \frac{\frac{m-1}{m}p_{\max}}{\frac{T}{m} - \frac{m-1}{m}p_{\max}} \leq 1 + \frac{\frac{m-1}{m}\cdot\frac{m-3}{(m-1)(m-2)}T}{\frac{T}{m} - \frac{m-1}{m}\cdot\frac{m-3}{(m-1)(m-2)}T} = m-2.$$

For $\frac{(m-3)}{(m-1)(m-2)}T < p_{\max} \leq T$, if $C^{H4} = L_m = p_{\max} > \frac{(m-3)}{(m-1)(m-2)}T$, we have

$$C^* \leq \frac{T}{m} < \frac{1}{m}\cdot\frac{(m-1)(m-2)}{(m-3)}p_{\max} \leq (m-2)p_{\max} = (m-2)C^{H4},$$

where the third inequality is valid when $m \geq 4$.

If $C^{H4} = \min_{1\leq i\leq m-1} L_i < L_m$, then $C^{H4}(\mathcal{M}, \mathcal{J}) = C^{H1}(\mathcal{M}', \mathcal{J}')$. In the following paragraph we will show that $C^*(\mathcal{M}, \mathcal{J}) \leq C^*(\mathcal{M}', \mathcal{J}')$. Therefore, by Theorem 2.2, we have

$$\frac{C^*(\mathcal{M}, \mathcal{J})}{C^{H4}(\mathcal{M}, \mathcal{J})} \leq \frac{C^*(\mathcal{M}', \mathcal{J}')}{C^{H1}(\mathcal{M}', \mathcal{J}')} \leq m' - 1 = m - 2.$$

Consider any optimal schedule S^* of $(\mathcal{M}, \mathcal{J})$. W.l.o.g., assume p_B is assigned to M_m in S^* together with jobs in $\mathcal{J}_0 \subseteq \mathcal{J}\backslash\{p_B\}$. Construct a feasible schedule S' of $(\mathcal{M}', \mathcal{J}')$ from S^* by moving jobs in $\mathcal{J}_0$ from M_m to M_{m-1}. Obviously, $L'_i \geq L^*_i$, $1 \leq i \leq m-1$, where L'_i and L^*_i are loads of M_i, $1 \leq i \leq m-1$ in S' and S^*, respectively. Therefore,

$$C^*(\mathcal{M}', \mathcal{J}') \geq \min_{1\leq i\leq m-1} L'_i \geq \min_{1\leq i\leq m-1} L^*_i \geq \min_{1\leq i\leq m} L^*_i = C^*(\mathcal{M}, \mathcal{J}). \quad \square$$

From Theorems 4.3 and 4.4, we know that $H4$ is the optimal algorithm for $Pm|sum\ \&\ max|C_{\min}$, $m \geq 4$, with competitive ratio $m-2$.

Acknowledgement

The authors would like to acknowledge the constructive comments by the anonymous referees which have improved the presentation of the paper. The first author's research was supported by the National Natural Science Foundation of China (10301028, 10671177, 60021201).

References

[1] Y. Azar, L. Epstein, On-line machine covering, Journal of Scheduling 1 (1998) 67–77.

[2] Y. Azar, O. Regev, On-line bin-stretching, Theoretical Computer Science 168 (2001) 17–41.
[3] T.C.E. Cheng, H. Kellerer, V. Kotov, Semi-on-line multiprocessor scheduling with given total processing time, Theoretical Computer Science 337 (2005) 134–146.
[4] J. Csirik, H. Kellerer, G. Woeginger, The exact LPT-bound for maximizing the minimum completion time, Operations Research Letters 11 (1992) 281–287.
[5] B. Deuermeyer, D. Friesen, M. Langston, Scheduling to maximize the minimum processor finish time in a multiprocessor system, SIAM Journal on Algebraic and Discrete Methods 3 (1982) 190–196.
[6] G. Dósa, Y. He, Semi-online agorithms for parallel machine scheduling problems, Computing 72 (2004) 355–363.
[7] L. Epstein, Bin stretching revisited, Acta Informatica 39 (2003) 97–117.
[8] Y. He, Semi on-line scheduling problem for maximizing the minimum machine completion time, Acta Mathematica Applicatae Sinica 17 (2001) 107–113.
[9] Y. He, The optimal on-line parallel machine scheduling, Computers & Mathematics with Applications 39 (2000) 117–121.
[10] Y. He, Z.Y. Tan, Randomized on-line and semi-on-line scheduling on identical machines, Asia-Pacific Journal of Operations Research 20 (2003) 31–40.
[11] Y. He, G.C. Zhang, Semi on-line scheduling on two identical machines, Computing 62 (1999) 179–187.
[12] H. Kellerer, V. Kotov, M.G. Speranza, Z. Tuza, Semi on-line algorithms for the partition problem, Operations Research Letters 21 (1997) 235–242.
[13] S. Seiden, J. Sgall, G. Woeginger, Semi-online scheduling with decreasing job sizes, Operations Research Letters 27 (2000) 215–227.
[14] Z.Y. Tan, Y. He, Semi-on-line problems on two identical machines with combined partial information, Operations Research Letters 30 (2002) 408–414.
[15] G. Woeginger, A polynomial time approximation scheme for maximizing the minimum machine completion time, Operations Research Letters 20 (1997) 149–154.